DOCUMENT RESUME

ED 475 937 IR 021 935

AUTHOR	King, Ronald S.; Rainwater, Stephen B.
TITLE	Linking Multiple Databases: Term Project Using "Sentences" DBMS.
PUB DATE	2002-06-00
NOTE	11p.; In: NECC 2002: National Educational Computing Conference Proceedings (23rd, San Antonio, Texas, June 17-19, 2002); see IR 021 916.
AVAILABLE FROM	For full text: http://confreg.uoregon.edu/necc2002/ .
PUB TYPE	Reports - Descriptive (141) -- Speeches/Meeting Papers (150)
EDRS PRICE	EDRS Price MF01/PC01 Plus Postage.
DESCRIPTORS	Computer Science Education; *Computer System Design; *Database Design; *Database Management Systems; Higher Education; Introductory Courses; Models; *Programming; Student Projects; World Wide Web

ABSTRACT

This paper describes a methodology for use in teaching an introductory Database Management System (DBMS) course. Students master basic database concepts through the use of a multiple component project implemented in both relational and associative data models. The associative data model is a new approach for designing multi-user, Web-enabled database operations. Unique capabilities of the associative data model include: omnicompetent programming, feature programming, instance schema processing, and schema aggregation. Potential applied research and application development as related to the associative data model and "Sentences," a multi-user, Web-enabled DBMS, are also addressed. (Author/AEF)

Reproductions supplied by EDRS are the best that can be made
from the original document.

Linking Multiple Databases: Term Project Using *Sentences* DBMS

PERMISSION TO REPRODUCE AND DISSEMINATE THIS MATERIAL HAS BEEN GRANTED BY

P.S. Calegari

TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

Ronald S. King and Stephen B. Rainwater
Department of Computer Science
The University of Texas at Tyler
Tyler, TX 75799
rking@mail.uttyl.edu

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

This document has been reproduced as received from the person or organization originating it.

Minor changes have been made to improve reproduction quality.

Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

ED 475 937

Abstract

This paper describes a methodology for use in teaching an introductory Database Management System course. Students master basic database concepts through the use of a multiple component project implemented in both relational and associative data models. The associative data model is a new approach for designing multi-user, web-enabled database operations. Unique capabilities of the associative data model include: omnicompetent programming, feature programming, instance schema processing, and schema aggregation. Potential applied research and application development as related to the associative data model and *Sentences* are also addressed.

Keywords

database design
associative data model
database course

1 Introduction

The database course can no longer concentrate solely on the relational data model. Clearly E-commerce is rapidly changing the whole realm of business. Information for E-commerce will come from data residing in a variety of internal and external sources, often accessed through the Web. At The University of Texas at Tyler, we teach an undergraduate DBMS class which focuses on the web as a front-end to a relational database and includes providing the details of the associative data model. Basically many of the guidelines in the course follow the recommendations set forth for the future of the first undergraduate database course [5] of a student's degree in information systems from computer science departments.

2 First Component of the Project

A term project involving merging databases and/or designing web-based databases is assigned to the students. A variation of the multiple-component project with associated reflection [4] is employed. In the first half of the course, databases are developed in ORACLE on a single server. Each student team (consisting of two to three students) is provided a secure tablespace on the server allowing access to tables directly or through Visual BASIC using ODBC drivers.

The database designed and implemented was for a mail-order business that purchases items from suppliers and fills orders from customers. Many active data features as well as some user-defined data types and functions were defined for a set of tables using constraints and triggers that encapsulated certain policies defining the operation of the business.

IR021935

BEST COPY AVAILABLE

With these components in mind, the first component of the project was to create the latter database. Students were asked to create new customers and add a few orders. During this component, each team had to implement their database design plus generate query and report sessions to demonstrate the functionality of their designs.

3 Second Component of the Project

At the completion of component 1, the teams were informed that their representative companies had merged. Each team was to retain their own identity but all reports and queries were to contain a unified set of data representing the aggregate of all merged companies. Additionally, each team was held responsible for the integrity of their company's database. An important feature was to avoid all duplication in the integrated database.

4 Third Component of the Project

The teams were asked to replicate components 1 and 2 utilizing the associative data model. Specifically, students were required to investigate the schema integration and instance schema processing features of the associative data model. Lazy Software's *Sentences* DBMS [6] was the DBMS employed for this component. *Sentences* is a multi-user, web-enabled DBMS written in Java complete with a full set of development tools, interfaces and applications. The first general release (1.1 *Sentences*), of the product was in October 2000 and runs under SUN Solaris, Windows NT or Linux. The ancillary tools that provide the user interface, schema specification facilities and query support can make use of Microsoft, Apache, WebSphere and Tomcat Web servers, with Windows clients running either Microsoft or Netscape browsers. At The University of Texas at Tyler, we are running a beta version of Release 2.0 (scheduled release fall 2001) which includes XML support, allowing for stored procedures and triggers.

The relational data model has many disadvantages which can be overcome by the associative data model. For instance, when employing the relational data model, each time a new application is constructed, the applications programmer would have to construct or modify a new set of tables. The latter practice is both time consuming and complex. Omnicompetent programming, a feature of the associative data model, allows for a single set of programs to implement many different applications. Also, companies or businesses often want to record information that is relevant only to a particular customer. In the relational data model, the programmer or database designer would accomplish the latter task via text or through null values in the relevant table column. Neither of these two solutions is truly satisfactory. Through the instance schema feature, the *Sentences* DBMS enables schema and rules to apply to a single instance in the database with no overhead. Additionally many company databases are reflecting the mergers taking place worldwide. In *Sentences*, distinct databases can be viewed together or amalgamated at any time without special tools for data correlation and analysis through the Schema aggregation feature. *Sentences* also allows for feature programming whereby features can be made visible or invisible to individual users. This latter feature is ideally suited to the needs of today's Application Service Providers (ASPs).

Sentences employs a set of technologies based around the Java2 platform including Java Servlets, JSP, and JDBC (Java Database Connectivity). Java Servlets are Web-enabled Java classes that can run on a Web server. Remote users connect to servlets using the same protocol as Common Gateway Interface

(CGI) scripts, which makes it easy to integrate servlets into the chosen Web architecture. JSP allows for the embedded Java code to be inserted directly into web pages, and it supports components called Java-Beans that hide complicated functionality behind a simple interface. Instead of updating Web pages every time programming logic changes are made, simple updates to the bean are completed and a copy of the bean is sent to the Web server. There isn't even a need to restart the server! JDBC is a set of libraries that connect Java programs, including Java servlets and JSPs to SQL databases.

Tomcat, the official reference implementation of Java servlets, enables the Web/database interfacing. Tomcat is an open source Apache Foundation project available at [1]. The Java2 SDK (java.sun.com/js2e) is required by Tomcat which is available for many operating systems including AIX, Linux, Mac OS X, Windows, and Solaris. Reference can be made to [3] on Tomcat installation.

5 Fourth Component of the Project

During this final phase of the project, the student teams were required to perform a series of system tests related to implementation of database conversion [2]. These system tests included:

- *Volume testing:* to verify that the system can handle the amount of data indicated by the specifications.
- *Stress testing*: to prove that the system performs correctly under peak loads; i.e., that it can process large quantities of data over a short period of time (especially important for web-enabled databases).
- *Performance testing*: to demonstrate that the system is able to process data as quickly as the specifications said it would.
- *Unit testing*: testing that concentrates on each unit (i.e. component) of the software as implemented in source code.
- *Integration testing:* where requirements established as part of software requirements analysis are validated against the software that has been constructed.
- *System testing:* where the software and other system elements are tested as a whole.
- *Acceptance testing*: to demonstrate that the global/ local databases met the business functions that were specified.

A comparative analysis of these various tests for the associative data model versus the relational data model were made by each student team.

6 The Associative Data Model

In the associative data model, a database comprises two types of data: entities and associations. Entities are things, or objects, that have discrete, independent existence. Associations are things whose existence depends on one or more other things, such that if any of those things ceases to exist, then the original thing itself ceases to exist or becomes meaningless. Associations are allowed to depend upon other associations. In the associative model, all attributes are represented as links between things within the database, and the target of every attribute is another thing that is represented within the database in its own right. In contrast, the relational data model has tuples as the source of an attribute and the target is the value contained by the cell under the column heading in the tuple. Thus in the associative data model, attributes are represented as: links between the entity or association whose attribute that is being

recorded as the source, a verb to express the nature of the attribute, and an entity or association as the target.

The impact of the latter representation is that things and associations are no different from associations in general. At any time the database designer or programmer may decide to describe an attribute by giving it attributes of its own. In the relational data model this would require restructuring the database, replacing a value by a foreign key and adding a new relation. The associative data model can be viewed as a vertically defined model on variable length data whereas the relational data model processes fixed length tuples horizontally.

6.1 Example *Sentences* Database

Consider the following mail order database application. Suppose the database designer needed to store the following piece of information: "Bob ordered a recordable compact disk on July 7, 2001 from Alpha, Inc." This piece of information contains four entities: Bob, recordable compact disk, date (July 7, 2001) and company (Alpha, Inc.). Additionally this information incorporates three verbs: ordered, on and from. Three links are required to store the data, as represented in the *Sentences* DBMS, by:

Bob ordered **recordable compact disk**
... on **July 7, 2001**
... from **Alpha, Inc.**

or by using parentheses we have:

(((Bob ordered recordable compact disk)
on July 7, 2001) from Alpha, Inc.)

This information could be implemented with following tables:

Items	
Identifier	**Name**
23	Bob
14	recordable compact disk
77	July 7, 2001
81	Alpha, Inc
92	ordered
101	on
16	from

Links			
Identifier	**Source**	**Verb**	**Target**
19	23	92	14
21	19	101	77
36	21	16	81

6.2 Relational Data Model for the Example *Sentences* Database

A relational data model for the mail-order business could include the following tables:

Customers [CustNo, Cname, Caddr, Balance, CreditLimit]

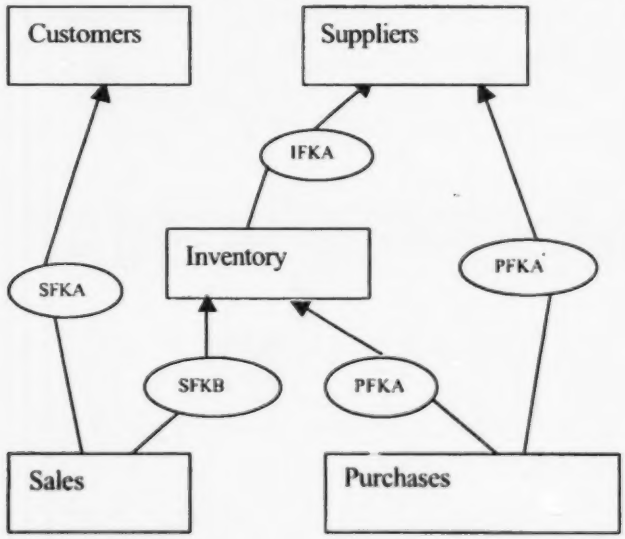
Suppliers [Sno, Sname, Saddr, AmtOwed]

Inventory [Ino, Iname, Sno, Qty-on-Hand, UnitPrice, Qty-on-Order, UnitOrderPrice, OrderThreshold, MinOrder]

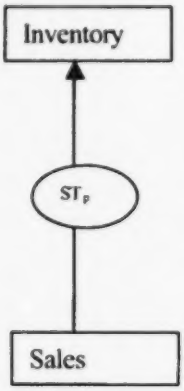
Purchases [OrderDate, OrdTime, Sno, Ino, QtyOrdered, DateRec, QtyRec, UnitPrice]

Sales [SaleDate, SaleTime, CustNo, Ino, QtySold, UnitPrice, TotalSale]

The following would represent a referential integrity diagram for this database where the notation XFKL is a foreign key from table X and L is a serial notation for all the foreign keys defined on table X.



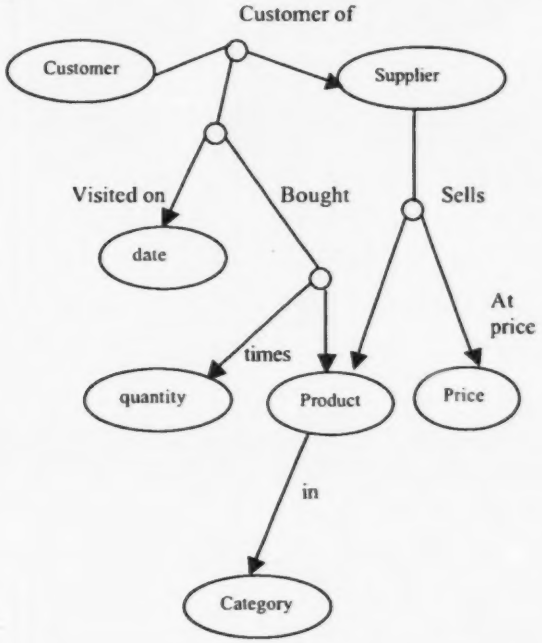
Finally various triggers can be defined on the latter database such as:



The pth trigger defined on Sales is ST_p. ST_p might be a trigger to increase the ordering threshold and possibly the minimum order quantity if the current value for the quantity sold for a given item in the last month is greater than the current threshold for that item. Triggers can be defined to maintain policy on: reordering supplies, what to do about customers who exceed their credit limit, cascading table inserts, maintaining audits, and other information deemed relevant.

6.3 Associative Data Model for the Example *Sentences* Database

A segment of a corresponding associative data model would involve items (ovals) and links (lines) in a semantic network diagram.



Next the semantic network is implemented in the *Sentences* DBMS. Every association has the following properties: a name, source type, target type, cardinality, inverse cardinality, being sequenced or sorted, and a default target. The associative data model permits an associative type that is specific to a single entity or association. Additionally, an entity or association type may be a subtype or supertype of another type. Besides subtypes and supertypes, the associative data model allows for the use of inferred subsets and supersets. For example:

Good Customer subset of **Customer**.

Basically there is a mapping between the relational and associative data models:

- A table with no foreign keys as primary keys is equivalent to an entity type that is the source of one or more association types.
- A domain is equivalent to an entity type that is the source of no association types.
- A table that has one or more foreign keys as primary keys is equivalent to an association type.
- The column heading of a type is equivalent to an association type's verb.

6.4 Query Language for the Associative Data Model

Associative algebra is the basis for query processing for the *Sentences* DBMS, which is derived directly from SQL. The following operators are available: union, intersection, difference, product, select, project, join, divide, extend, summarize, rename, and recursive closure. Only extend, summarize, and recursive closure differ from the traditional SQL operators. Extend forms the associative type that has the original type as source and a new type, instances of which are derived from the source as target. Summarize forms a type whose instances are formed by grouping instances of the original type that have the same sub-tree as source, and creating one instance of the new type for each subgroup, with the sub-tree as source and an instance aggregating corresponding sub-trees of the group as target. The recursive closure forms a relation by joining a self-referencing type with itself, taking the result and joining it again with the original type, as many times as necessary.

The schema for the latter associative data model for the mail-order business would be:

Customer customer of **Supplier**
… visited on **date**
… bought **Product**
… time **Quantity**
Supplier sells **Product**
… at **Price**
Product in **Category**

Sample data would include:

Bob customer of **Best Buy**
… visited on **25-June-01**
… bought **ink cartridge**
… times **10**
… bought **diskettes**
… times **200**

Bill customer of **Ables Land**
… visited on **6-July-01**
… bought **office chair**
… times **3**
… bought **transparency**
… times **10**
… bought **file folders**
… times **3**
… bought **diskettes**
… times **150**
Best Buy sells **ink cartridges**
… price **$39**
Best Buy sells **diskettes**
… price **$3**
Best Buy sells **big manila envelope**
… price **$10**

Best Buy sells **correction pen**
... price **$2.75**
Ables Land sells **office chair**
... price **$70**
Ables Land sells **transparency**
... price **$10**
Ables Land sells **file folde r**
... price **$20.69**
file folder category **office supply**
office chair category **furniture**
ink cartridge category **computer accessory**
diskette category **computer accessory**
transparency category **film**
big manila envelope category **office supply**
correction pen category **office supply**

Example queries in the *Sentences* DBMS are:

"Who shops at Best Buy?"
Q: Select (**Customer** customer of **"Best Buy"**)
A: **Bob** customer of **Best Buy**

"What computer facilities has Bill bought?"
Q: Select (((**"Bill"** customer of **Supplier**) visited on **date**) bought **Product**) join (**Product** category **"computer facility"**))
A: (((**Bill** customer of **Ables Land**) visited on **6-July-01**) bought **diskettes**) join (**diskettes** category **computer facilities**))

7 Schema Integration

Database integration involves the process by which information from participating databases can be conceptually integrated to form a single cohesive definition of a multi-database. The design process in multi-database systems is bottom-up. In other words, the individual databases actually exist; this is followed by the design of the global conceptual schema integrating these component databases into a multi-database. Under phase one of the process, schema translation, the designer faces the task of mapping one schema to another. For n-ary relational databases the integrator must make trade-offs between the different local schemas to determine which representation should be given precedence when conflicts arise. This requires that the integrator have knowledge of all the various trade-offs that must be made among several different schemas and their semantics, which may be different. Phase two, schema integration, involves the process of: *identifying* the components of a database which are related to one another, *selecting* the best representation for the global conceptual schema, and finally, *integrating* the components of each intermediate schema. Foreign keys, preservation of the lossless-join property, and preservation of the functional dependency preserving property are among a short list of the constraints that make schema translation and schema integration for the n-ary relational data model difficult. Sentences implementation of the associative model of data with the concepts of profiles and chapters plus the surrogate links for representing entities and associations make schema integration efficient as well as simple.

8 Potential Applied Research and Application Development

Sentences, being the first commercial implementation of the associative model of data, needs many enhancements. These enhancements offer opportunities for applied research and application developments in the following areas:

- Methodologies for triggers and constraints to implement business policy
- How to enable concurrency
- Development of migration tools for moving between the n-ary relational data model and the associative model of data
- Establishing benchmarks
- Development of transaction processing algorithms
- Determination of whether or not *Sentences* will facilitate a "large" data warehouse
- How to implement temporal databases utilizing the associative model of data

Currently TPC (the Transaction Processing Council) benchmarks enable performance evaluations which are available for the n-ary relational and object-relational data models. Even a web-based version, TPC-W, is available. But these benchmarks are unacceptable for performance evaluations on the associative data model since transactions are not operationally defined in terms of atomic table operations such as joins, selections, and projections. The associative model of data implemented in *Sentences* is based upon query trees.

Migration studies could be realized through XML. *Sentences* allows for both importing and exporting the data in the database, the schema, and the query trees. The relational data model can be implemented by XML Schema, a data definition language for XML documents. Data for the relational model can be represented using XML. Since XPath views XML documents as trees and element attributes, comments, and text as nodes of those trees, the investigator would have the ability to develop mappings between the traditional SQL queries and the query trees of the associative model of data in *Sentences*.

XML Spy 3.5, an integrated visual development environment for XML, could be used to complete the migration study. The tool includes XML editing and validation, Schema/DTD editing and validation, and XML editing and transformation. A demonstration version of the software is available on the Web at xmlspy.com/xml-editor.html. A fully functional 30-day trial version is available for download. Turbo XML, a Java-based IDE for developing and managing XML assets, is also available as a free download on the Web from TIBCO Extensibility.

Triggers and constraints in version 2.0 of *Sentences* require the user to implement business policy in Java Jars and register them with the kernel of the database. A major contribution by an investigator would be to develop the windows event-driven triggers that would mirror the trigger capabilities similar to those found in products like ORACLE or DB2.

Concurrency control appears to be a major issue with *Sentences* since the model is implemented using only two pseudo-tables. *Sentences* allows for the source of an association to be another association. Therefore concurrency implementation for *Sentences* should be based upon the foundation of nested and multilevel transactions.

9 Conclusion

At The University of Texas at Tyler, the core material normally presented in an undergraduate database class is enhanced by having students provided with an extensive hands-on experience in database design, implementation, operations, and maintenance. By having the team project involve multi-user, web-enabled applications such as the integration of multiple databases into a single large database, students are exposed to a firm foundation for further studies and participate in an experience which is invaluable in the *real-world* environment.

The *Sentences* DBMS has been very well received by students in the undergraduate course at The University of Texas at Tyler. *Sentences* is easy to use, simple in design features yet contains most of the features that a database instructor would want to illustrate as limitations of relational and object-relational DBMSs.

References

[1] Apache Foundation Project. Available: http://jakarta.apache.org/tomcat.

[2] Babb, V. G. The Analysis, Design and Implementation of a Database Conversion. *ACM SIGCSE Bulletin*, Vol. 20, No. 2, pp. 60-64, June, 1988.

[3] Goodwills, J. Tomcat Installation. Available: http://www.oreilleynet.com/pub/ct/33.

[4] Robert, M. A. Enhancing the Value of a Project in the Database Course. *Proceedings of 31st SIGCSE Symposium* (March, 2000), ACM Press, 36.

[5] Springsteel, F., Robert, M. A., and Ricardo, C. M. The Next Decade of the Database Course: Three Decades Speak to the Next. *Proceedings of 31st SIGCSE Symposium* (March, 2000), ACM Press, 41.

[6] Williams, S. *The Associative Data Model*, Lazy Software Ltd., 2000.

U.S. Department of Education
Office of Educational Research and Improvement (OERI)
National Library of Education (NLE)
Educational Resources Information Center (ERIC)

NOTICE

Reproduction Basis

X	This document is covered by a signed "Reproduction Release (Blanket)" form (on file within the ERIC system), encompassing all or classes of documents from its source organization and, therefore, does not require a "Specific Document" Release form.
	This document is Federally-funded, or carries its own permission to reproduce, or is otherwise in the public domain and, therefore, may be reproduced by ERIC without a signed Reproduction Release form (either "Specific Document" or "Blanket").